Contents

1 Project overview

2 Beremiz's user manual
   2.1 Beremiz installation .................................................. 5
   2.2 Start a new automation project ........................................ 6
   2.3 Write your own POU .................................................... 6
   2.4 Build PLC executable binary ........................................... 6
   2.5 Beremiz and Beremiz_service connectors ............................ 6
   2.6 Trace POUs instances variable ....................................... 8

3 IEC 61131-3 ........................................................................ 9

4 PLCopen TC6 ..................................................................... 11
Contents:
In order to target the widest possible range of programmable devices and keep efficient, Beremiz use C code as an intermediate language.

To be executed, C needs to be compiled. GCC serve that purpose perfectly.

PLC program is expressed in languages defined in IEC-61131, including graphical languages. Thanks to PLCopen TC2, those graphical languages have a standardised representation, in XML.

To be continued.
2.1 Beremiz installation

2.1.1 Windows

Download installer, install.

2.1.2 Linux

Pre-requisites:

```bash
# Ubuntu/Debian :
sudo apt-get install python-wxgtk2.8 pyro mercurial
sudo apt-get install build-essential bison flex python-numpy python-nevow
```

Prepare:

```bash
mkdir ~/Beremiz
cd ~/Beremiz
```

Get Source Code:

```bash
cd ~/Beremiz
hg clone http://dev.automforge.net/beremiz
hg clone http://dev.automforge.net/plcopeneditor
hg clone http://dev.automforge.net/matiec
```

Build MatIEC compiler:
cd ~/Beremiz/matiec
./configure
make

Build CanFestival (optional):

# Only needed for CANopen support. Please read CanFestival manual to choose CAN interface other than 'virtual':

cd ~/Beremiz
hg clone http://dev.automforge.net/CanFestival-3

cd ~/Beremiz/CanFestival-3
./configure --can=virtual
make

Launch Beremiz:

cd ~/Beremiz/beremiz
python Beremiz.py

2.2 Start a new automation project

2.3 Write your own POUs

2.4 Build PLC executable binary

2.5 Beremiz and Beremiz_service connectors

To connect a PLC, Beremiz provides 2 types of connectors:

- a Pyro connector
- a WAMP connector

To configure the connection, you have to set the URI_location in your project Config tab according to this documentation.

2.5.1 The Pyro connector

Pyro is an advanced and powerful Distributed Object Technology system written entirely in Python. Beremiz_service spawns a Pyro server, serving a PLCObject (see runtime/PLCObject.py). Therefore, Beremiz acts as a Pyro client.

TODO:: link to PLCObject API documentation

URI_location:

- LOCAL:// is a facility that starts the PLC service locally and connect Beremiz to it via Pyro. This is intended for use in development stage.
- PYRO://<ip:port> normal connection to a remote PLC. PLC default port is 3000.
- PYROS://<ip:port> SSL connection to a remote PLC, see below.
more information about Pyro can be found on http://pythonhosted.org//Pyro/1-intro.html

**Setup a Pyro SSL connection**

Pyro v3 has a limited TLS/SSL support based on m2crypto. Pyro v4 had dropped it. In order to have a full and reliable SSL, we recommand to use a TLS/SSL wrapper as nginx, stub or stunnel.

**TLS-PSK with stunnel**

In this example, we setup a simple TLS-PSK connection according to rfc4279. This ciphersuite avoid the need for public key operations and certificate management. It is perfect for a performance-constrained environments with limited CPU power as a PLC.

**Needed**:

- stunnel >= 5.09

verify openssl support for PSK cipher:

```
openssl ciphers -v 'PSK'
```

**Client setup (Beremiz)**

You need to choose an identity for your client, here `client1`. generate a valid and strong key:

```
$ echo client1:$(openssl rand -base64 48) > pskclient1.txt
```

write a stunnel client configuration file `stunnel-client.conf`:

```
output = stunnel-client.log
client = yes

[beremiz]
accept = 3002
connect = [PLC]:3001
PSKidentity = client1
PSKsecrets = pskclient1.txt
```

start stunnel client side:

```
stunnel stunnel-client.conf
```

You could now connect beremiz with classic URI: `location = PYRO://127.0.0.1:3002`

**Server setup (PLC)**

import the client key in a keyfile psk.txt, concatenating all client key.

write a stunnel server configuration file `stunnel-server.conf`:
output = stunnel-server.log

[beremiz]
accept = 3001
connect = 127.0.0.1:3000
PSKsecrets = psk.txt

start stunnel server side:

\texttt{stunnel \ stunnel-server.conf}

more documentation on \texttt{stunnel} \url{http://www.stunnel.org/docs.html}

\subsection{2.5.2 The WAMP connector}

WAMP is an open standard WebSocket subprotocol that provides two application messaging patterns in one unified protocol: Remote Procedure Calls + Publish & Subscribe.

Beremiz WAMP connector implementation uses Autobahn and crossbar.

\textbf{URI location}:
\begin{itemize}
  \item WAMP://127.0.0.1:8888#Automation#2534667845
\end{itemize}

more information about WAMP can be found on \url{http://wamp.ws/}

\subsection{2.6 Trace POUs instances variable}
IEC-61131-3

IEC-61131 is a normative document provided by the standards organization IEC (International Electrotechnical Commission) and describing a standard for implementing programmable controllers.

The part 3 of this document (commonly named IEC 61131-3) specifies syntax and semantics for programming language for programmable controllers. Beremiz implements all the languages described in this document.

http://www.iec.eu
PLCopen is a vendor- and product-independent worldwide association defining international standards for various topics related to control programming. For this purpose, PLCopen has 6 technical committees.

The goal of the sixth committee (TC6) is to define a standard file format, based on XML, for exchanging programmables controllers programmed using IEC 61131-3 languages. Beremiz uses this file format for saving the PLC programs of projects.

http://www.plcopen.org